CS380I Programming Assignment 2: LiveOak - 2 to SaM Compiler

1. Assignment Overview

1.1 Task

Create a handwritten recursive-descent parser and SaM code generator for levels 0, 1, and 2 of the LiveOak language. Use the provided SaMTokenizer class for lexical analysis. The compiler should take a file containing a Live e containing a SaM program that is a correct tra aims to help understand recursive-descent parsing a ated lecture videos include LiveOak, stack machines an eneration.

1.2 Score and Du

The score is 100 points. Th n use up to 2 late (extension) days.

1.3 Submission M

All submissions are electron . The submission testing is automated on Gradescope file of your project) and source.zip (a.zip file containing all your source files).

2. Preparation Materials

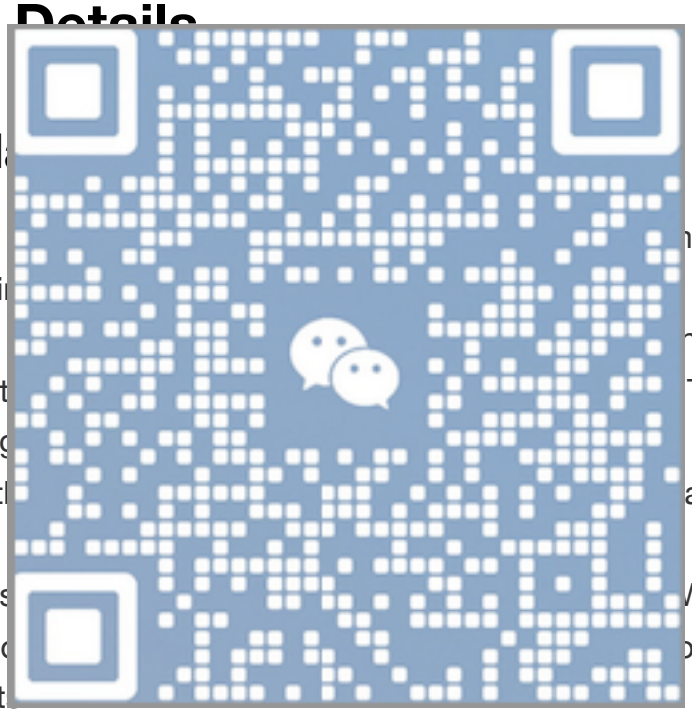
2.1 Required Materials

- The SaM library (v2.6.3) containing the lexer (obtained from Programming Assignment 1). Compile your code with this.jar file. If you use an IDE, you need to add this.jar file as an external library to your project.
- A collection of public test cases (still being expanded).

2.2 Optional Materials

- A starter template for your compiler, available here (https://utexas.instructure.com/courses/1404625/ files/79509966?wrap=1) (https://utexas.instructure.com/courses/1404625/files/79509966/ download?download_frd=1).
- The HTML documentation of the SaM API, available here (https://utexas.instructure.com/courses/1404625/files/79016316?wrap=1) (https://utexas.instructure.com/courses/1404625/files/79016316/download?download_frd=1).
- The design document containing the complete workings of the SaM interpreter, available here (https://utexas.instructure.com/courses/1404625/files/79016315?wrap=1) (https://utexas.instructure.com/courses/1404625/files/79016315/download?download_frd=1).

3. Assignment Details



3.1 Grammar Rel

- There will always be a n starts executing from the main method. The mai
- There will always be a nput program.
- Comments in the input Tokenizer class.
- There is no overloadin
- The definition of a met ach method needs a symbol table.
- A break statement mus When executed, it terminates the executi ou are required to handle illegal break statement
- If a program does not satisfy the grammar or the textual description of the language, the compiler should print a short error message and/or exit with a non-zero exit status.

3.2 Compiler Related

The compiler should be in the Java class assignment2.LiveOak2Compiler. It should take two command-line arguments. The first is an input file containing a LiveOak program. The second is an output file that will contain the generated SaM code. It is recommended to handle complexity by working up through the levels of the source language.

4. Evaluation

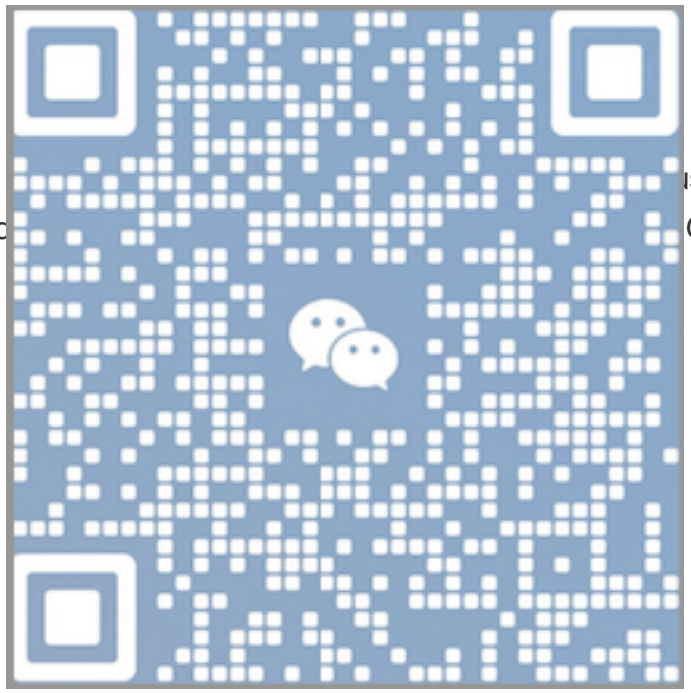
4.1 Evaluation Commands

- Compiling a LiveOak - 2 program: java -jar compiler.jar test1.lo output.sam
- Running a SaM program: java -cp SaM - 2.6.3.jar edu.utexas.cs.sam.ui.SamText output.sam

4.2 Evaluation Criteria

The compiler's correctness is evaluated based on the program's exit status. The public test cases are not exhaustive. It is highly recommended to create more test cases for testing. Before submitting the.jar file, make sure the exit status matches the expected output on all test cases.

5. Resources

The assignment counts for  using Gradescope. The names of the test cases inc Grading is based only on the LiveOak - 2 test cases.